Bees and Firefly Algorithms for Noisy Non-Linear Optimisation Problems

N. Chai-ead, P. Aungkulanon*, and P. Luangpaiboon, *Member, IAENG*

Abstract — **Effective methods for solving the complex and noisy engineering problems using a finite sequence of instructions can be categorised into optimisation and meta-heuristics algorithms. The latter might be defined as an iterative search process that efficiently performs the exploration and exploitation in the solution space aiming to efficiently find near optimal solutions. Various natural intelligences and inspirations have been adopted into the iterative process. In this work, two types of meta-heuristics called Bees and Firefly algorithms were adapted to find optimal solutions of noisy non-linear continuous mathematical models. Considering the solution space in a specified region, some models contain global optimum and multiple local optimums. Bees algorithm is an optimisation algorithm inspired by the natural foraging behaviour of honey bees. Firefly algorithm is used to produce a near optimal solution under a consideration of the flashing characteristics of fireflies. A series of computational experiments using each algorithm were conducted. Experimental results were analysed in terms of best solutions found so far, mean and standard deviation on both the actual yields and execution time to converge to the optimum. The Firefly algorithm seems to be better when the noise levels increase. The Bees algorithm provides the better levels of computation time and the speed of convergence. In summary, the Firefly algorithm is more suitable to exploit a search space by improving individuals' experience and simultaneously obtaining a population of local optimal solutions.**

Index Terms — **Meta-Heuristic, Bees Algorithm, Firefly Algorithm, Noisy Non-linear Optimisation**

I. Introduction

The optimisation of systems and processes is very meaning to the efficiency and economics of many science and engineering domains. Optimisation problems are solved by using rigorous or approximate mathematical search techniques. Rigorous methods have employed linear programming, integer programming, dynamic programming or branch-and-bound techniques to approach the optimal solution for moderate-size problems. However, optimising real-life problems of the scale often encountered in engineering practice is much more challenging because of the huge, complex and noisy solution space. Finding exact solutions to these problems turn out to be NP-hard. This kind of complex problem requires an exponential amount of computing power and time, as the number of decision variables increases. To overcome these problems, researchers have proposed approximate evolutionary-based or meta-heuristics algorithms as a means to search for near-optimal solutions [1].

Generally, meta-heuristics work as follows: a population of individuals is randomly initialised where each individual represents a potential solution to the problem. The quality of each solution is then evaluated via a fitness function. A selection process is applied during the iteration of meta-heuristics in order to form a new population. The searching process is biased toward the better individuals to increase their chances of being included in the new population. This procedure is repeated until convergence rules are reached.

Learning from life system, people have developed many optimisation computation approaches to solve complicated problems in the recent decades. Researchers have developed computational systems that mimic the efficient behaviour of species such as ants, bees, birds and frogs, as a means to seek faster and more robust solutions to complex and noisy optimisation problems. The evolutionary based techniques introduced in the literature were Genetic Algorithm or GA [2], Memetics Algorithm or MAs [2], Shuffled Frog Leaping Algorithm or SFLA [2], Firefly Algorithm or FFA [3, 4, 5], Bees Algorithm or BEES [6, 7],Harmony Search Algorithm or HSA [8], Neural Network or NN [9], Ant Colony Optimisation or ACO [10], Evolutionary Programming or EP [11], Differential Evolution or DE [12] and Particle Swarm Optimisation or PSO [13]. Moreover, there are some with the socially-based inspiration, e.g. Taboo Search or TS [14] and the physically-based inspiration such as Simulated Annealing or SA [15]. These algorithms have been widely used in many industrial and social areas. These kinds of algorithms for scientific computation are called as ''Artificial-Life Computation".

A relatively new branch of nature inspired meta-heuristics which are called as swarm intelligence is focused on insect behaviour in order to mimic insect's problem solution abilities. Interaction between insects contributing to the collective intelligence of the social insect colonies is focused. A new population-based search algorithm called the Bees Algorithm (BEES) was then presented. The algorithm mimics the food foraging behaviour of swarms of honey bees. A colony of honey bees can extend itself over long distances (more than 10 kilometres) and in multiple directions simultaneously to exploit a large number of food sources. A colony prospers by deploying its foragers to good fields. One of the examples of such interactive behaviour is the waggle dance of honey bees during the food harvesting. By performing this at the dance floor, successful foragers share the useful information about the direction and distance

Manuscript received December 9, 2010; revised January 24, 2011. This work was supported in part the Thailand Research Fund (TRF), the National Research Council of Thailand (NRCT) and the Commission on Higher Education of Thailand. The authors wish to thank the Faculty of Engineering, Thammasat University, THAILAND for the financial support.

N. Chai-ead is a master student with the Industrial Statistics and Operational Research Unit (ISO-RU), Department of Industrial Engineering, Faculty of Engineering, Thammasat University, 12120, THAILAND.

*P. Aungkulanon is a Ph. D. Candidate, ISO-RU, Department of Industrial Engineering, Faculty of Engineering, Thammasat University, 12120, THAILAND. [Phone: (662)564-3002-9; Fax: (662)564-3017; e-mail: pasurachacha@hotmail.com, lpongch@engr.tu.ac.th].

P. Luangpaiboon is an Associate Professor, ISO-RU, Department of Industrial Engineering, Faculty of Engineering, Thammasat University, 12120, THAILAND.

to patches of flower and the amount of nectar within this flower with their hive mates. This is a successful mechanism which foragers can recruit other bees in their colony to productive patches. Bee colony can efficiently and precisely adjust its searching pattern in time and space according to changing nectar sources.

The other meta-heuristic algorithm, which idealises some of the flashing characteristics of fireflies, has been recently developed and named the Firefly algorithm (FFA). Nature-inspired methodologies are currently among the most powerful algorithms for optimisation problems. FFA is a novel nature-inspired algorithm inspired by social behavior of fireflies. Fireflies are one of the most special, captivating and fascinating creature in the nature. There are about two thousand firefly species, and most fireflies produce short and rhythmic flashes. The rate and the rhythmic flash, and the amount of time form part of the signal system which brings both sexes together. Therefore, the main part of a firefly's flash is to act as a signal system to attract other fireflies. By idealising some of the flashing characteristics of fireflies, the firefly-inspired algorithm was presented by Xin-She Yang [3]

The objective of this paper is to investigate the performance of Firefly and Bees algorithm to find optimal solutions of noisy unconstrained mathematical models with continuous design variables. Various standard benchmark engineering optimisation examples from the literature are also presented to demonstrate the effectiveness and robustness of the meta-heuristics. This paper is organised as follows. Section II describes the selected meta-heuristic of Bees algorithm including its pseudo code. Sections III describes the selected meta-heuristic of Firefly algorithm. Section IV and V are briefing about tested models and computational results and analyses, respectively. The conclusion is also summarised and it is followed by acknowledgment and references.

II. Bees Algorithm (BEES)

A. *Bees in Nature*

A colony of honey bees can be seen as a diffuse creature which can extend itself over long distances in various directions in order to simultaneously exploit a large number of food sources [6, 7]. In principle, flower patches with plentiful amounts of nectar or pollen that can be collected with less effort should be visited by more bees, whereas patches with less nectar or pollen should receive fewer bees.

The foraging process begins in a colony by scout bees being sent to survey for promising flower patches. Scout bees search randomly from one patch to another. A colony of honey bees can extend itself over long distances in multiple directions of a search space. During the harvesting season, a colony continues its exploration, keeping a percentage of the population as scout bees. When they return to the hive, those scout bees that found a patch which is rated above a certain threshold (measured as a combination of some constituents, such as sugar content) deposit their nectar or pollen and go to the “dance floor” to perform a dance known as the “waggle dance”.

This dance is essential for colony communication, and contains three vital pieces of information regarding a flower patch: the direction in which it will be found, its distance from the hive or energy usage and its nectar quality rating (or fitness). This information helps the bees to find the flower patches precisely, without using guides or maps.

Each individual’s knowledge of the outside environment is gleaned solely from the waggle dance. This dance enables the colony to evaluate the relative merit of different patches according to both the quality of the food they provide and the amount of energy needed to harvest it. After waggle dancing on the dance floor, the dancer bee (i.e. the scout bee) goes back to the flower patch with follower bees that were waiting inside the hive. The number of follower bees assigned to a patch depends on the overall quality of the patch.

This allows the colony to gather food quickly and efficiently. While harvesting from a patch, the bees monitor its food level. This is necessary to decide upon the next waggle dance when they return to the hive. If the patch is still good enough as a food source, then it will be advertised in the waggle dance and more bees will be recruited to that source.

B. *Bees Algorithm*

Bees Algorithm is an optimisation algorithm inspired by the natural foraging behaviour of honey bees [3, 4]. Fig. 1 shows the pseudo code for the algorithm in its simplest form. The algorithm requires various influential parameters to be preset, namely: the number of scout bees (*n*), the number of patches selected out of *n* visited points (*m*), the number of elite patches out of *m* selected patches (*e*), the number of bees recruited for the best *e* patches (*nep*), the number of bees recruited for the other (*m*-*e*) selected patches (*nsp*) and the size of patches (*ngh*) including stopping criterion.

The algorithm starts with the *n* scout bees being randomly placed in the search space of feasible solutions. The fitnesses of the points visited by the scout bees are evaluated in the second step. Step 3, the scout bees are classified into various groups. In step 4, bees that have the highest fitnesses are designated as “selected bees” and sites visited by them are chosen for neighbourhood search. Then, in steps 5 and 6, the algorithm conducts searches in the neighbourhood of the selected bees, assigning more bees to search near to the best *e* bees.

The bees can be chosen directly according to the fitnesses associated with the points they are visiting. Alternatively, the fitness values are used to determine the probability of the bees being selected. Searches in the neighbourhood of the best *e* bees which represent more promising solutions are made more detailed by recruiting more bees to follow them than the other selected bees. Together with scouting, this differential recruitment is a key operation of the Bees Algorithm. In step 6, for each site only the bee with the highest fitness will be selected to form the next bee population. In nature, there is no such a restriction. This constraint is introduced here to reduce the number of points to be explored. In step 7, the remaining bees in the population are assigned randomly around the search space scouting for new potential solutions. These steps are repeated until a stopping criterion is met. At the end in each iteration, the colony will have two parts to its new population – representatives from each selected patch and other scout bees assigned to conduct random searches. The Bee dance function to provide the related useful information for finding the food is followed:

$$x_{i+1} = (x_i\text{-}ngh) + (2*ngh*rand\left(0-1\right)*(upper\ x_i\text{-}lower\ x_i)$$

The algorithm has been successfully applied to different problems including of neural network optimisations, training pattern recognition, scheduled jobs for a machine, data clustering and tuning the fuzzy logic controller. Fig. 1 shows the pseudo code for the BEES in its simplest form.

```
Procedure BEES Meta-heuristic()
Begin;
  Initialise algorithm parameters:
     n:    the number of scout bees
     m:    the number of sites selected out of n visited sites
     e:    the number of the best sites out of m selected sites
     nep:  the number of bees recruited for the best e sites,
     nsp:  the number of bees recruited for the other m-e selected sites
     ngh:  the initial size of patches
  Randomly initialise the bee population;
  Evaluate fitnesses of the bee population;
  While (stopping criterion not met)
     Form the new bee population;
     Select sites for neighbourhood search;
     Recruit bees for selected sites with more bees for better e sites;
     Evaluate the fitnesses;
  End while;
End procedure;
```

Fig. 1. Pseudo code of the BEES Meta-heuristic.

III. Firefly algorithm (FFA)

A. *Firefly in Nature*

Fireflies or glowworms are the creatures that can generate light inside of it. Light production in fireflies is due to a type of chemical reaction. This process occurs in specialised light-emitting organs, usually on a firefly's lower abdomen. It is thought that light in adult fireflies was originally used for similar warning purposes, but evolved for use in mate or sexual selection via a variety of ways to communicate with mates in courtships. Although they have many mechanisms, the interesting issues are what they do for any communication to find food and to protect themselves from enemy hunters including their successful reproduction.

The pattern of flashes is often unique for a particular species of fireflies. The flashing light is generated by a chemical process of bioluminescence. However, two fundamental functions of such flashes are to attract mating partners or communication, and to attract potential victim. Additionally, flashing may also serve as a protective warning mechanism. Both sexes of fireflies are brought together via the rhythmic flash, the rate of flashing and the amount of time form part of the signal system. Females respond to a male’s unique pattern of flashing in the same species, while in some species, female fireflies can mimic the mating flashing pattern of other species so as to lure and eat the male fireflies who may mistake the flashes as a potential suitable mate.

The light intensity at a particular distance from the light source follows the inverse square law. That is as the distance increases the light intensity decreases. Furthermore, the air absorbs light which becomes weaker and weaker as there is an increase of the distance. There are two combined factors that make most fireflies visible only to a limited distance that is usually good enough for fireflies to communicate each other. The flashing light can be formulated in such a way that it is associated with the objective function to be optimised. This makes it possible to formulate new meta-heuristic algorithms.

B. *Firefly Algorithm*

The firefly algorithm (FFA) is a meta-heuristic algorithm, inspired by the flashing behaviour of fireflies. The primary purpose for a firefly's flash is to act as a signal system to attract other fireflies. Now this can idealise some of the flashing characteristics of fireflies so as to consequently develop firefly-inspired algorithms. For simplicity in describing our new Firefly Algorithm (FFA) [3, 4], there are the following three idealised rules.

On the first rule, each firefly attracts all the other fireflies with weaker flashes [16]. All fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex. Secondly, attractiveness is proportional to their brightness which is reverse proportional to their distances. For any two flashing fireflies, the less bright one will move towards the brighter one. The attractiveness is proportional to the brightness and they both decrease as their distance increases. If there is no brighter one than a particular firefly, it will move randomly. Finally, no firefly can attract the brightest firefly and it moves randomly.

The brightness of a firefly is affected or determined by the landscape of the objective function. For a maximisation problem, the brightness can simply be proportional to the value of the objective function. Other forms of brightness can be defined in a similar way to the fitness function in genetic algorithms. Based on these three rules, the basic steps of the firefly algorithm (FFA) can be summarised as the pseudo code shown in Fig. 2.

```
Procedure FFA Meta-heuristic()
Begin;
  Initialise algorithm parameters:
     MaxGen:   the maximal number of generations
     $\gamma$:        the light absorption coefficient
     r:        the particular distance from the light source
     d:        the domain space
  Define the objective function of f($\mathbf{x}$), where $\mathbf{x}=(x_1,........,x_d)^T$
  Generate the initial population of fireflies or $\mathbf{x}_i$ (i=1, 2 ,..., n)
  Determine the light intensity of $I_i$ at $\mathbf{x}_i$ via f($\mathbf{x}_i$)
  While (t<MaxGen)
    For i = 1 to n (all n fireflies);
       For j=1 to n (n fireflies)
             if ($I_j > I_i$), move firefly i towards j; end if
             Attractiveness varies with distance r via Exp [$-\gamma r^2$];
             Evaluate new solutions and update light intensity;
       End for j;
    End for i;
   Rank the fireflies and find the current best;
  End while;
  Postprocess results and visualisation;
End procedure;
```

Fig. 2. Pseudo code of of the FFA Meta-heuristic.

In the firefly algorithm there are two important issues of the variation of light intensity and the formulation of the attractiveness. For simplicity, it is assumed that the attractiveness of a firefly is determined by its brightness which in turn is associated with the encoded objective function of the optimisation problems. On the attractiveness of the FFA the main form of attractiveness function or β(*r*) can be any monotonically decreasing functions such as the following generalised form of

$$\beta(r) = \beta_0 e^{-\gamma r^m}, \ (m \geq 1),$$

where *r* or r_{ij} is the distance between the *i*th and *j*th of two fireflies. β_0 is the attractiveness at $r = 0$ and γ is a fixed light absorption coefficient. The distance between any two fireflies *i* and *j* at $\mathbf{x}_i$ and $\mathbf{x}_j$ is the Cartesian distance as follows:

$$r_{ij} = \left\| x_i - x_j \right\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \ ,$$

where x_{ik} is the *k*-th component of the *i*-th firefly ($\mathbf{x}_i$). The movement of a firefly, *i* is attracted to another more attractive (brighter) firefly *j*, is determined by

$$x_{i+1} = x_i + \beta_0 e^{-\gamma r_{ij}^2} (x_j - x_i) + \alpha(rand - 0.5) \ ,$$

where the second term is due to the attraction while the third term is the randomisation with α being the randomisation parameter. Rand is a random number generator uniformly distributed in the range of [0, 1]. For most cases in the implementation, $\beta_0 = 1$ and $\alpha = [0, 1]$. Furthermore, the randomisation term can easily be extended to a normal distribution *N* (0, 1) or other distributions.

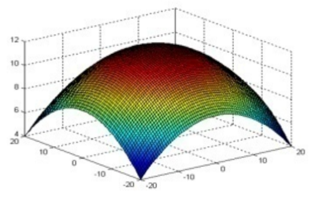
Additionally, if the scales vary significantly in different dimensions such as −105 to 105 in one dimension while, say, −0.001 to 0.01 along the other, it is a good idea to replace α by αS_k where the scaling parameters S_k (k = 1, ..., d) in the *d* dimensions should be determined by the actual scales of the problem of interest. The parameter γ characterises the variation of the attractiveness, and its value is crucially important in determining the speed of the convergence and how the FFA behaves. In most applications, it typically varies from 0.01 to 100.

IV. Tested Models

In this paper, the algorithms operate and analyse the results under various type of continuous mathematical functions with two variables. The comparison is made with the measurement noise, normally and independently distributed with zero mean and standard deviation of 0, 1, 2 and 3, on the process yields. The typical three-dimensional response surfaces are shown in Figures A-H.

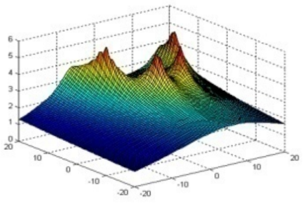
The typical natures of selected surfaces to be used in this study are the single peak of parabolic function, the multi-peak of Camelback, Rastrigin and Shekel functions and the curved ridge of Rosenbrock and Styblinski functions including the multi-peak with curved ridge of Branin Goldstein-Price functions. However, there is the limitation of merely 2-variable problems.

A. Parabolic Function



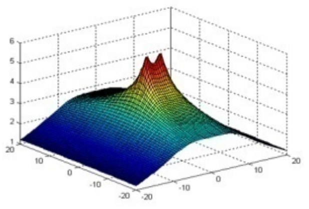
$$\text{Max } f(x_1, x_2) = 12 - (x_1^2 + x_2^2/100)$$

B. Branin Function



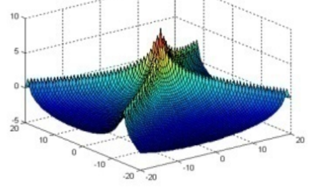
$$\text{Max } f(x_1, x_2) = 5 - \log_{10}[(x_2 - (5.1/4\pi^2)\,x_1^2 + ((5/\pi)\,x_1 - 6)^2 + ((10 - (5/4\pi))\cos(x_1)) + 10]$$

C. Camelback Function



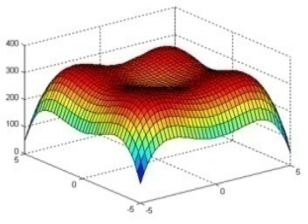
$$\text{Max } f(x_1, x_2) = 10 - \log_{10}[x_1^2 - (4 - 2.1x_1^2 + (1/3)\,x_1^4) + x_1 x_2 + 4\,x_2^2(x_2^2 - 1)]$$

D. Goldstein-Price Function



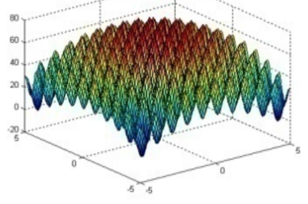
$$\text{Max } f(x_1, x_2) = 10 + \log_{10}[1/\{(1 + (1 + x_1 + x_2)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2))*(30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))\}]$$

E. Styblinski Function



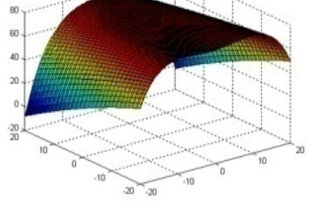
$$\text{Max } f(x_1, x_2) = 275 - [\,((x_1^4 - 16x_1^2 + 5x_1)/2) + ((x_2^4 - 16x_2^2 + 5x_2)/2) + 3\,]$$

F. Rastrigin Function



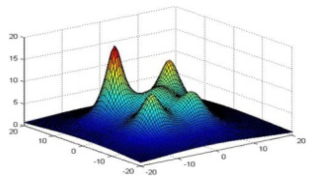
$$\text{Max } f(x_1, x_2) = 80 - [20 + x_1^2 + x_2^2 - 10(\cos(2\pi\,x_1) + \cos(2\pi\,x_2))]$$

G. Rosenbrock Function



$$\text{Max } f(x_1, x_2) = 70\ [[[\,20 - \{(1 - x_1/-7)^2 + ((x_2/6) + (x_1/-7)^2)^2\}] + 150]/170] + 10$$

H. Shekel Function



$$\text{Max } f(x_1,x_2) = 100\ [1/(9+(x_1-4)^2+(x_2-6)^2)+ 1/(20+(x_1+0)^2+(x_2-0)^2)+1/(14+(x_1-8)^2+(x_2+3)^2) +1/(11+(x_1-8)^2+(x_2-8)^2)+1/(6+(x_1+6)^2+(x_2-7)^2)$$

V. Computational Results And Analyses

In this work, for the computational procedures described above a computer simulation program was implemented in a Visual C#2008 computer program. A Laptop computer Aspire Z99Sseries ASUS was used for computational experiments throughout. A numerical comparison of the conventional procedures of the Bees and Firefly algorithms are determined in this section. These meta-heuristics were adapted to search optimal solutions of non-linear mathematical models without constraints. Considering the solution space in a specified region of response surfaces, some models contain global optimum and multiple local optimums as described above.

FFA and BEES algorithms are optimisation algorithms inspired by the natural foraging behavior of honey bees and social behavior of fireflies and the phenomenon of bioluminescent communication, respectively. They are the meta-heuristics with the similar naturally-based inspiration which include Particle Swarm Optimisation (PSO) or Artificial Bee Colony (ABC) techniques. Experimental results involved a performance comparison of the FFA and BEES algorithms under a limitation of the 2-variable problems.

Each algorithm has its own influential parameters that affect its performance in terms of solution quality and execution time. To achieve the most preferable parameter choices that suit the tested problems, a large number of experiments were conducted. For each algorithm, an initial setting of the parameters was established using values previously reported in the literature. Then, the parameter values were developed via the experimental designs and the results were monitored in terms of various solution quality measures. The final parameter values adopted in each algorithm are followed and will be applied for all optimisation problems presented in this paper.

BEES parameters were set as follows: the number of scout bees (n) = 50, the number of sites selected out of n visited sites (m) = 10, the number of best sites out of m selected sites (e) = 5, the number of bees recruited for best e sites (nep) = 5, the number of bees recruited for the other m-e selected site (nsp) = 10, the initial size of patches (ngh) = 0.1. FFA parameters were set as follows: $\beta_0 = 1$, $\alpha = [0, 1]$, $\gamma = [0.01, 100]$ and the number of fireflies = 40. Both algorithms were executed with the same designed points of 6000 realisations. There are fifteen trial runs in each problem and noise level. The performance of the different algorithms was compared using three criteria which comprise of the mean and standard deviation of actual process yields and the processing time to reach the optimum at the maximal preset design points.

When there was no noise on the process yields, the performance of both algorithms of the BEES and FFA seems to be not different to approach to the optimum. The average and standard deviation (STD) of actual yields and the computation time (Tables I and II) including maximal and minimal actual yields achieved by the FFA tend to be better, especially on the multi-peak functions, when the standard deviation of noises (N) raise from 1 to 3 (Fig. 3). Moreover, the consistency of the FFA performs quite well that could be indicated by the standard deviation of yields from 15 replications.

Complexity or difficulty of the functions had no effect to the FFA as expected except Camelback function. However, execute time in each replication is dramatically higher when compared, especially on the functions with curved ridge or mixed curved ridge and multi-peak natures. BEES seems to be better in terms of speed of convergence (Fig. 4 and 5). This might be the effect from generating the completely different random numbers to use in the iterative procedures of the algorithm.

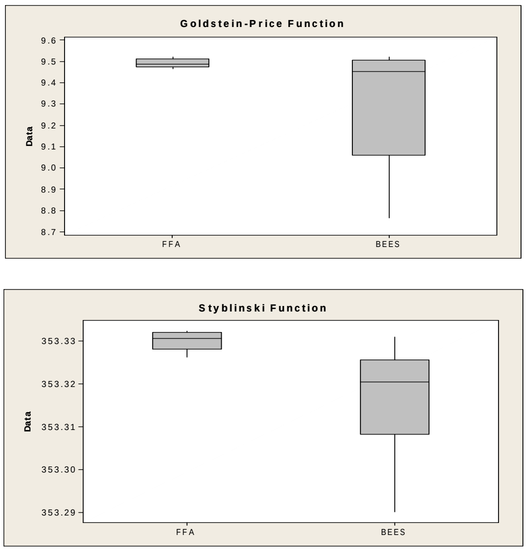


Fig. 3. Graphical Results on Goldstein-Price and Styblinski Function.

This implies that the FFA is more potentially powerful in solving noisy non-linear optimisation problems. The FFA seems to be a promising optimisation tool in part due to the effect of the attractiveness function which is a unique of firefly behaviour. The FFA has not only the self improvement process with the current space, but it also includes the improvement among its own space from the previous stages whereas the BEES provides only the procedure of bee dance improvement. As also appeared on the Particle Swarm Optimisation (PSO), this leads the proper level of convergence to the optimum.

References

[1] E. Elbeltag, T. Hegazy and D. Griersona, “Modified Shuffled Frog-Leaping Optimisation Algorithm: Applications to Project Management”, *Structure and Infrastructure Engineering*, vol. 3, no. 1, 2007, pp. 53 – 60.

[2] E. Emad, H. Tarek, and G. Donald, “Comparison among Five Evolutionary-based Optimisation Algorithms”, *Advanced Engineering Informatics*, vol.19, 2005, pp. 43-53.

[3] X.S. Yang, “A Discrete Firefly Meta-heuristic with Local Search for Make span Minimisation in Permutation Flow Shop Scheduling Problems”, *International Journal of Industrial Engineering Computations*, vol. 1, 2010, pp. 1–10.

[4] X.S. Yang, “Firefly Algorithms for Multimodal Optimisation”, *Stochastic Algorithms: Foundations and Applications*, SAGA 2009, Lecture Notes in Computer Sciences, vol. 5792, 2009, pp. 169-178.

[5] S. Lukasik and S. Zak, “Firefly Algorithm for Continuous Constrained Optimisation Tasks”, *Systems Research Institute, Polish Academy of Sciences*, 2010, pp. 1–10.

[6] D.T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim and M. Zaidi, “The Bees Algorithm. Technical Note”. *Manufacturing Engineering Centre, Cardiff University, UK*, 2005.

[7] D.T. Pham, Ghanbarzadeh A., Koc E., Otri S., Rahim S., and M. Zaidi, “The Bees Algorithm - A Novel Tool for Complex Optimisation Problems", *Proceedings of IPROMS 2006 Conference*, 2006, pp. 454-461.

[8] K.S. Lee and Z.W. Geem, “A New Meta-heuristic Algorithm for Continues Engineering Optimisation: Harmony Search Theory and Practice”, *Comput: Meth. Appl. Mech. Eng.,* vol. 194, 2004, pp. 3902–3933.

[9] P. Muller and D.R. Insua, "Issues in Bayesian Analysis of Neural Network Models", *Neural Computation*, vol. 10, 1995, pp. 571–592.

[10] M. Dorigo, V. Maniezzo and A. Colorni, “Ant System: Optimisation by a Colony of Cooperating Agents”, *IEEE Transactions on Systems, Man, and Cybernetics Part B*, vol. 26, numéro 1, 1996, pp. 29-41.

[11] J.Y. Jeon, J.H. Kim, and K. Koh “Experimental Evolutionary Programming-based High-precision Control,” *IEEE Control Sys. Tech.*, vol. 17, 1997, pp. 66-74.

[12] R. Storn, “System Design by Constraint Adaptation and Differential Evolution", *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 1, 1999, pp. 22-34.

[13] M. Clerc, and J. Kennedy, “The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space”, *IEEE Transactions on Evolutionary Computation*, vol. 6, 2002, pp.58-73.

[14] Lokketangen, A. K. Jornsten and S. Storoy “Tabu Search within a Pivot and Complement Framework”, *International Transactions in Operations Research*, vol. 1, no. 3, 1994, pp. 305-316.

[15] V. Granville, M. Krivanek and J.P. Rasson, “Simulated Annealing: a Proof of Convergence”, *Pattern Analysis and Machine Intelligence, IEEE Transactions,* vol. 16, Issue 6, 1994, pp. 652 – 656.

[16] H. Zang, S. Zhang and K. Hapeshi, ”A Review of Nature-Inspired Algorithms”, *Journal of Bionic Engineering,* vol. 7, 2010, pp. 232–237.

TABLE I
EXPERIMENTAL RESULTS OBTAINED FROM THE BEES ON EACH TESTED FUNCTION

Tested Function		N=0	Time	N=1	Time
Branin	Mean	5.40	281.21	4.80	281.10
	STD.	0.00	0.30	0.48	0.02
Camelback	Mean	12.89	281.13	11.10	281.09
	STD.	0.80	0.14	1.13	0.00
Gold S.P.	Mean	9.28	281.13	8.69	281.13
	STD.	0.27	0.13	0.70	0.11
Parabolic	Mean	12.00	281.12	11.90	281.09
	STD.	0.00	0.09	0.12	0.02
Rastrigin	Mean	99.35	281.17	98.63	281.12
	STD.	1.02	0.14	1.52	0.10
Rosenbrock	Mean	80.00	281.22	79.84	281.09
	STD.	0.00	0.37	0.12	0.02
Shekel	Mean	18.98	281.10	18.63	281.12
	STD.	0.00	0.01	0.27	0.12
Styblinski	Mean	353.32	281.14	352.65	281.12
	STD.	0.01	0.09	0.48	0.07
Tested Function		N=2	Time	N=3	Time
Branin	Mean	3.52	281.18	3.22	281.096
	STD.	0.52	0.12	0.58	0.002
Camelback	Mean	10.02	281.09	9.31	281.135
	STD.	1.51	0.01	0.87	0.151
Gold S.P.	Mean	8.06	281.11	7.27	281.095
	STD.	0.70	0.07	0.93	0.001
Parabolic	Mean	11.46	281.09	11.19	281.147
	STD.	0.53	0.00	0.55	0.200
Rastrigin	Mean	98.26	281.16	97.6	281.133
	STD.	1.87	0.29	1.95	0.144
Rosenbrock	Mean	79.71	281.13	79.72	281.095
	STD.	0.27	0.17	0.24	0.001
Shekel	Mean	18.08	281.11	17.71	281.096
	STD.	0.83	0.04	0.819	0.002
Styblinski	Mean	351.65	281.14	351.53	281.120
	STD.	1.36	0.08	1.346	0.094

TABLE II
EXPERIMENTAL RESULTS OBTAINED FROM THE FFA ON EACH TESTED FUNCTION

Tested Function		N=0	Time	N=1	Time
Branin	Mean	5.400	660.124	4.108	2282.105
	STD.	0.000	0.277	0.371	4388.727
Camelback	Mean	13.014	660.083	9.854	662.378
	STD.	0.285	0.126	1.049	4.808
Gold S.P.	Mean	9.492	660.089	6.161	662.008
	STD.	0.020	0.183	0.354	4.722
Parabolic	Mean	12.000	660.073	11.913	660.803
	STD.	0.000	0.106	0.011	1.934
Rastrigin	Mean	99.989	689.861	99.513	661.399
	STD.	0.010	115.386	1.062	3.718
Rosenbrock	Mean	80.000	660.053	79.931	660.879
	STD.	0.000	0.049	0.028	1.840
Shekel	Mean	18.980	689.829	18.976	660.285
	STD.	0.000	115.364	0.004	0.938
Styblinski	Mean	353.330	660.083	353.321	660.923
	STD.	0.002	0.112	0.009	2.347
Tested Function		N=2	Time	N=3	Time
Branin	Mean	3.902	1283.567	3.728	659.958
	STD.	0.812	2414.949	0.618	0.143
Camelback	Mean	8.414	660.028	7.994	659.952
	STD.	0.684	0.070	0.304	0.159
Gold S.P.	Mean	5.820	660.044	5.867	659.948
	STD.	0.370	0.094	0.435	0.167
Parabolic	Mean	11.915	660.040	11.896	659.973
	STD.	0.008	0.118	0.031	0.104
Rastrigin	Mean	95.724	1283.538	95.432	659.989
	STD.	0.268	2414.876	0.613	0.020
Rosenbrock	Mean	79.933	660.048	79.940	659.960
	STD.	0.014	0.134	0.004	0.188
Shekel	Mean	18.962	660.025	18.954	659.984
	STD.	0.014	0.054	0.020	0.002
Styblinski	Mean	353.301	660.040	353.280	659.966
	STD.	0.023	0.089	0.045	0.102

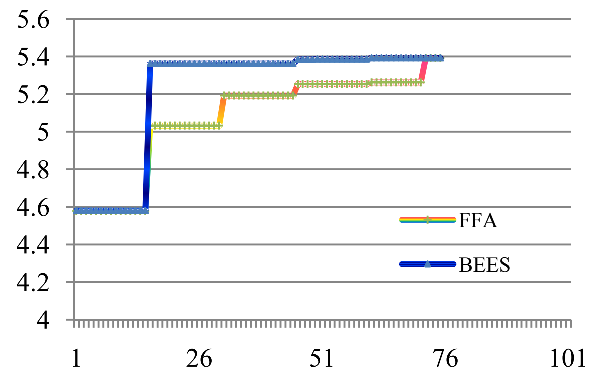


Fig. 4. Speed of Convergence on Branin Function.

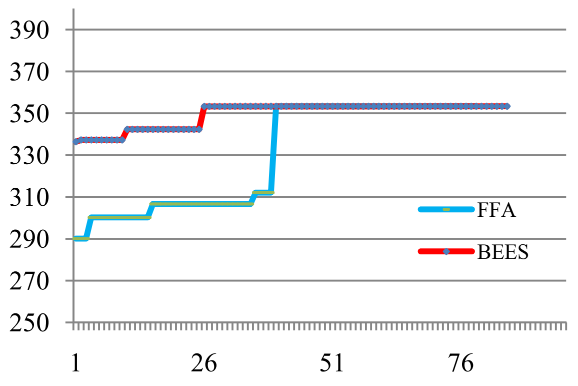


Fig. 5. Speed of Convergence on Styblinski Function.